* OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

Object-oriented programming has several advantages over procedural programming:

* OOP is faster and easier to execute
* OOP provides a clear structure for the programs
* OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
* OOP makes it possible to create full reusable applications with less code and shorter development time
* Why And When To Use "Inheritance" and "Polymorphism"?

It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class.

* Why Encapsulation? 🡪 Using get and set

Better control of class attributes and methods

Class attributes can be made read-only (if you only use the get method), or write-only (if you only use the set method)

Flexible: the programmer can change one part of the code without affecting other parts

Increased security of data

* The final Keyword

If you don't want other classes to inherit from a class, use the final keyword.

* Abstract

An abstract method belongs to an abstract class, and it does not have a body. The body is provided by the subclass.

* Why And When To Use Abstract Classes and Methods?

To achieve security - hide certain details and only show the important details of an object.

* Static

A static method means that it can be accessed without creating an object of the class, unlike public.

* Constructors

Note that the constructor name must match the class name, and it cannot have a return type (like void).

Also note that the constructor is called when the object is created.

All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you. However, then you are not able to set initial values for object attributes.

* Constructor Parameters

Constructors can also take parameters, which is used to initialize attributes.

The following example adds an int y parameter to the constructor. Inside the constructor we set x to y (x=y). When we call the constructor, we pass a parameter to the constructor (5), which will set the value of x to 5.

* Java Inner Classes

In Java, it is also possible to nest classes (a class within a class). The purpose of nested classes is to group classes that belong together, which makes your code more readable and maintainable.

To access the inner class, create an object of the outer class, and then create an object of the inner class.

* Interfaces

Another way to achieve abstraction in Java, is with interfaces.

An interface is a completely "abstract class" that is used to group related methods with empty bodies.

To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the implements keyword (instead of extends). The body of the interface method is provided by the "implement" class.

* Notes on Interfaces

Like abstract classes, interfaces cannot be used to create objects (in the example above, it is not possible to create an "Animal" object in the MyMainClass)

Interface methods do not have a body - the body is provided by the "implement" class

On implementation of an interface, you must override all of its methods

Interface methods are by default abstract and public

Interface attributes are by default public, static and final

An interface cannot contain a constructor (as it cannot be used to create objects)

* Why And When To Use Interfaces?

To achieve security - hide certain details and only show the important details of an object (interface).

Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because the class can implement multiple interfaces. Note: To implement multiple interfaces, separate them with a comma (see example below).

* Enums

An enum is a special "class" that represents a group of constants (unchangeable variables, like final variables).

To create an enum, use the enum keyword (instead of class or interface), and separate the constants with a comma. Note that they should be in uppercase letters.

* Difference between Enums and Classes

An enum can, just like a class, have attributes and methods. The only difference is that enum constants are public, static and final (unchangeable - cannot be overridden).

An enum cannot be used to create objects, and it cannot extend other classes (but it can implement interfaces).

* Why And When To Use Enums?

Use enums when you have values that you know aren't going to change, like month days, days, colors, deck of cards, etc.

* Java User Input

The Scanner class is used to get user input, and it is found in the java.util package.

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings.

* Java ArrayList

The ArrayList class is a resizable array, which can be found in the java.util package.

The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want. The syntax is also slightly different:

* Java LinkedList

In the previous chapter, you learned about the ArrayList class. The LinkedList class is almost identical to the ArrayList.

Use an ArrayList for storing and accessing data, and LinkedList to manipulate data.

The LinkedList class is a collection which can contain many objects of the same type, just like the ArrayList.

The LinkedList class has all of the same methods as the ArrayList class because they both implement the List interface. This means that you can add items, change items, remove items and clear the list in the same way.

However, while the ArrayList class and the LinkedList class can be used in the same way, they are built very differently.

* Java HashMap

In the ArrayList chapter, you learned that Arrays store items as an ordered collection, and you have to access them with an index number (int type). A HashMap however, store items in "key/value" pairs, and you can access them by an index of another type (e.g. a String).

One object is used as a key (index) to another object (value). It can store different types: String keys and Integer values, or the same type, like: String keys and String values.

* Java Iterator

An Iterator is an object that can be used to loop through collections, like ArrayList and HashSet. It is called an "iterator" because "iterating" is the technical term for looping.

To use an Iterator, you must import it from the java.util package.

* Java Exceptions

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).